REMARKS

Applicant's representative expresses appreciation for the in person interview conducted on April 9. The amendments made by this response and the remarks are consistent with that which was discussed during the interview.

The Office Action mailed December 31, 2008, considered and rejected claims 1-4, 7, 10-17 and 20-27. Claims1-4, 7, 10-17 and 20-27 were rejected under 35 U.S.C. § 102(b) as being anticipated by the TETware Released 3.3 software product (hereinafter TETware) released September 18, 1998 by The Open Group, as evidenced by: "TETware User Guide, Revision 1.2" (hereinafter TET_UG), Release Notes for TETware Release 3.3" (hereinafter TET_RN), and "TETware Programmers Guide, Revision 1.2" (hereinafter TET_PG).[1]

By this response, claims 1, 3, 4, 15, and 24 are amended, while claim 2 is canceled. Support for these amendments may be found on page 13, lines 10-20, and pages 14-16.[2] Claims 1-4, 7, 10-17, and 20-27 remain pending of which claims 1, 4, 15, and 24 are independent.

The present invention is directed to embodiments for providing a generic test harness that can run tests that are written in any language or format. One problem which the present invention seeks to resolve is the proliferation of test harnesses that are written for testing programs. For example, a typical programming team writes its own harness or harnesses to execute tests in its own proprietary test format. Much time is spent writing and maintaining these harnesses. The present invention overcomes this problem by defining a generic harness that can execute test cases using a language and format independent interface. In essence, the invention is compatible with any type of test case because it provides a layer of abstraction between the test case and the harness. The harness treats any test as a black box – it doesn't know anything about the specific details of the test, and the test knows nothing about the harness that will execute it. The harness defines an abstract interface (i.e. "a language and format independent interface") which the connector must expose in the test case hierarchy thus allowing the harness to read and execute the test cases "regardless of the language or format in which the one or more available test cases were written." *See* Claim 1. As an example, this abstraction

[1] Although the prior art status of the cited art is not being challenged at this time, Applicant reserves the right to challenge the prior art status of the cited art at any appropriate time, should it arise. Accordingly, any arguments and amendments made herein should not be construed as acquiescing to any prior art status of the cited art.

[2] The independent claims have been further amended as discussed during the interview to better define the use of the COM interface. This interface is introduced beginning on page 14, lines 12-14.

using a language and format independent interface may be achieved using COM technology. However, the independent claims should not be construed as narrowly, as the invention could also be implemented using any architecture that defines a means for accessing resources over a network, such as the .NET framework. *See* Pg. 13, lines 18-20.

The amendments to the independent claims have been made to better clarify this abstraction that is obtained by implementing a language and format independent interface. For example, the amendments state that the connector includes the independent interface within the test case hierarchy. The amendments also emphasize that the harness uses the independent interface to access the test cases in the hierarchy. The cited references do not disclose or suggest this type of abstraction as claimed.

TETware, for example, uses a specific API for each test case depending on the language used to write the test case. In TETware, the Test Case Controller (tcc) executes the test cases and would therefore be similar to the harness of the present invention. Also, a Test Case Manager (TCM) is used to supervise this execution and performs the linking of the test code to the API library to produce executable test cases. *See* PG 2.4.2. As such the TCM is the most similar component of TETware to the connector in the present invention. However, TETware defines "a separate TCM module for each API that is supported." PG 2.4.2. These separate modules are shown in section 2.4 of the User Guide. As is shown, there is a separate Test Case Manager for code written in C, C++, Shell, Korn, and Perl. UG 2.4. In other words, TETware does not abstract the language and format of the test case from the harness. Any test case that a user wishes to test on TETware must conform to one of the TCMs provided. As such this scenario cannot teach or suggest the limitations of the independent claims which each require the harness to execute the test cases through a language and format independent interface.

TETware is also able to execute non API-conforming test cases. *See* PG 2.4.4. "A test case which does not use a TETware API is known as a non API-conforming test case." PG 2.4.1. If a test case is non-conforming, "tcc assumes that the test case writes journal output to stdout and stderr and regards the whole execution as if it were a single invocable component containing a single test purpose." PG 2.4.4. In other words, when a non conforming test is run, TETware simply records the output of the test case to a file. Therefore, in this scenario, no interface of TETware is used to execute the test case. As such this scenario cannot teach or

suggest the limitations of the independent claims which each require the harness to execute the test cases through a language and format independent interface.

Additionally, the independent claims require that the language and format independent interface is incorporated into the test case hierarchy (i.e. the connector exposes the interface defined by the harness). In TETware, the API-conforming test cases are coded to a language specific API rather than including a language and format independent interface. As such, TETware fails to teach or suggest this limitation of each of the independent claims.

Finally, although the directory structure of the test suite may be considered hierarchical (e.g. each test suite must provide at least one test scenario file which includes the individual test cases), the hierarchical structure is not built by the connector. In contrast, in TETware, the user must create the test suite to have at least one test scenario file. *See* PG 2.5.2, pg 7. TETware simply takes the test scenario as input and processes each included test case in sequence. *See* PG 4.4 (showing three test cases in Example 1 that are executed in order). The present invention is different in that the processing begins with a program module containing the test cases which are then searched by the connector to find the test cases. The test cases are then organized into the hierarchy to which the independent interface is added. TETware discloses none of these steps that are performed by the connector to create the hierarchy. As such, TETware fails to teach or suggest these limitations of the independent claims.

Hartmann was also cited to reject dependent claim 3 that is related to the use of COM technology. However, Hartmann likewise fails to teach or suggest the abstraction aspect as addressed above. Hartmann discloses embodiments for testing COM objects. *See* Col. 1, lines 36-40, 55-58; Col. 2, lines 61-64; and Col. 3, lines 3-4. This is done by generating a state machine representation of the COM objects using UML. *See* Col. 13, lines 35-40. ITL test cases are then generated from the state machine representation. *See* Col. 15, lines 13-18. The ITL test cases are then mapped to object oriented code. Finally, the object oriented code is compiled to generate the executable test cases. *See* Col. 30, lines 28-36. For a succinct listing of these steps, see claim 1 in column 34. As can be seen, the state machine representation, the ITL test cases, the object oriented code, and the executable test cases are all specific to a language. There is no abstraction involved in Hartmann. There is nothing similar to the language and format independent interface that is used by the harness to execute the test cases. In fact, the test cases

are compiled into a specific object oriented language. Therefore, the combination of TETware and Hartmann fails to render the independent claims obvious.

In view of the foregoing, Applicant respectfully submits that the other rejections to the claims are now moot and do not, therefore, need to be addressed individually at this time. It will be appreciated, however, that this should not be construed as Applicant acquiescing to any of the purported teachings or assertions made in the last action regarding the cited art or the pending application, including any official notice. Instead, Applicant reserves the right to challenge any of the purported teachings or assertions made in the last action at any appropriate time in the future, should the need arise. Furthermore, to the extent that the Examiner has relied on any Official Notice, explicitly or implicitly, Applicant specifically requests that the Examiner provide references supporting the teachings officially noticed, as well as the required motivation or suggestion to combine the relied upon notice with the other art of record.

In the event that the Examiner finds remaining impediment to a prompt allowance of this application that may be clarified through a telephone interview, the Examiner is requested to contact the undersigned attorney at (801) 533-9800.

Dated this 30th day of April, 2009.

Respectfully submitted,

Brian Tucker

RICK D. NYDEGGER
Registration No. 28,651
BRIAN D. TUCKER
Registration No. 61,550
Attorneys for Applicant
Customer No. 47973

RDN:BDT:gd
2304116_1